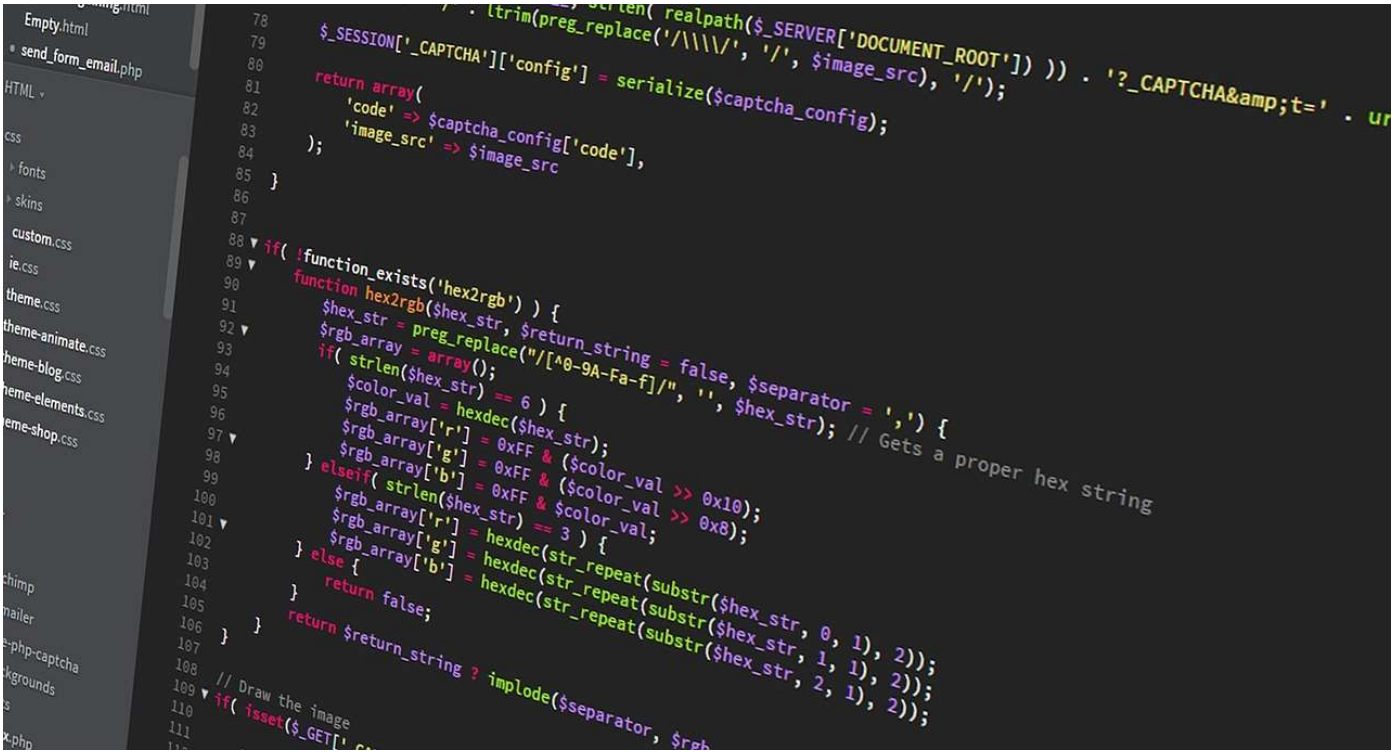
ARTÍCULOS DE TECNOLOGÍA

Funciones en javaScript

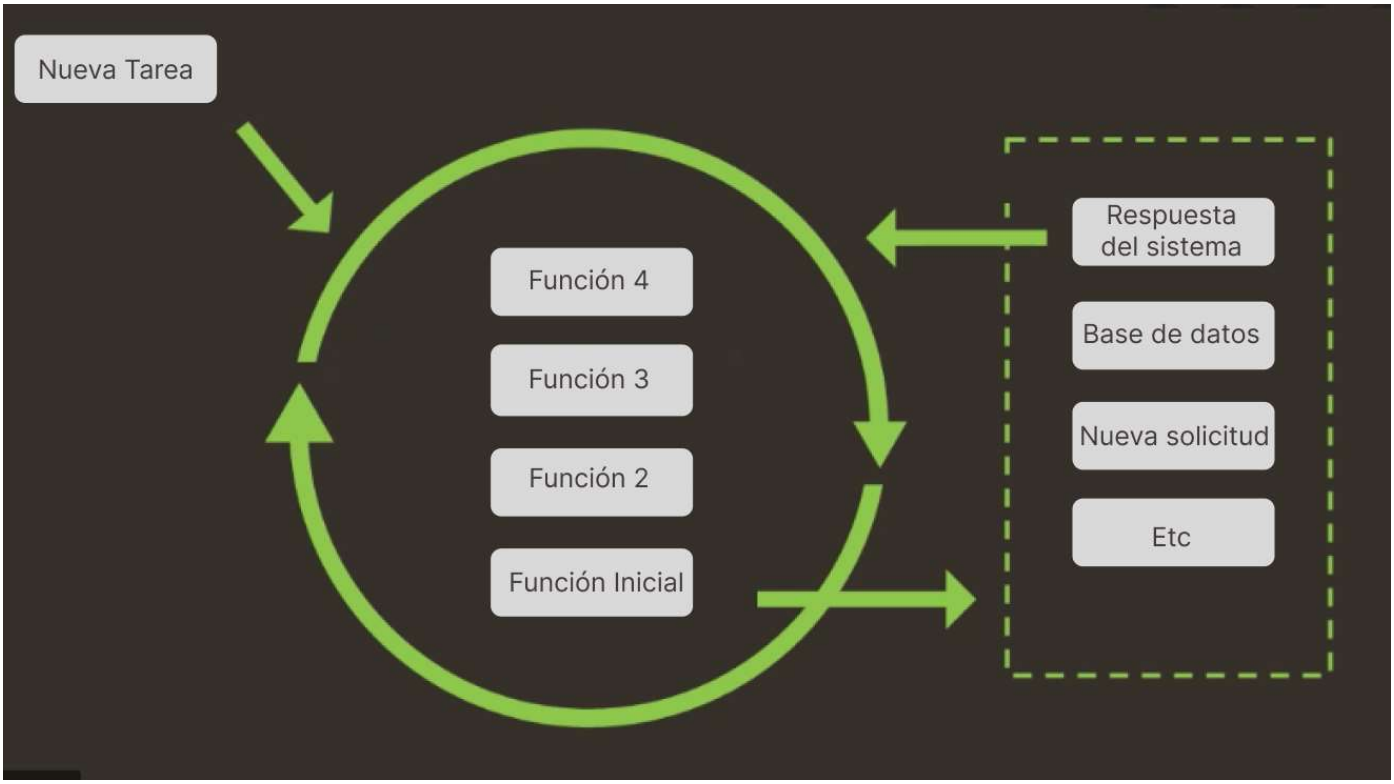
Marianna Costa
02/09/2022



¿Qué son las funciones?

Las funciones son un conjunto de instrucciones que realizan una tarea. En JavaScript, las funciones son fundamentales para la construcción de un código. Además, se consideran "objetos de primera clase" porque pueden tener propiedades y métodos como cualquier otro objeto.

Una de las formas de aprovechar por completo este poderoso lenguaje, seria saber **¿cómo Javascript entiende una función?**



En el flujo anterior tenemos un ejemplo al que llamamos repeticiones de eventos (Event Loop). Que de forma simplificada, es como, cada una de las acciones del código son interpretadas por el navegador al ejecutar un código JavaScript.

Imagine ser un usuario que ha abierto una página y empieza a interactuar con ella: escribe algo, hace clic en links, mientras se carga algún contenido. Estas son las funciones iniciales que desencadenan otras acciones con el tiempo, al paso que interactúa con la página. Cada una de estas acciones genera solicitudes que el servidor puede almacenar y devolver en respuesta, a la vez que se pueden agregar nuevas tareas.

Lo ideal aquí es entender que todas las funciones generan acciones que van a ser desencadeadas una tras otra, y estos comandos son precisamente las funciones que escribe un desarrollador al construir el código.

Conociendo este flujo, podemos entender más sobre estas estructuras que hacen de la web lo que conocemos y usamos en nuestra vida diaria: **las funciones**.

Existen varios tipos de funciones, todas con propósitos diferentes, a continuación aprenderemos más sobre las siguientes funciones:

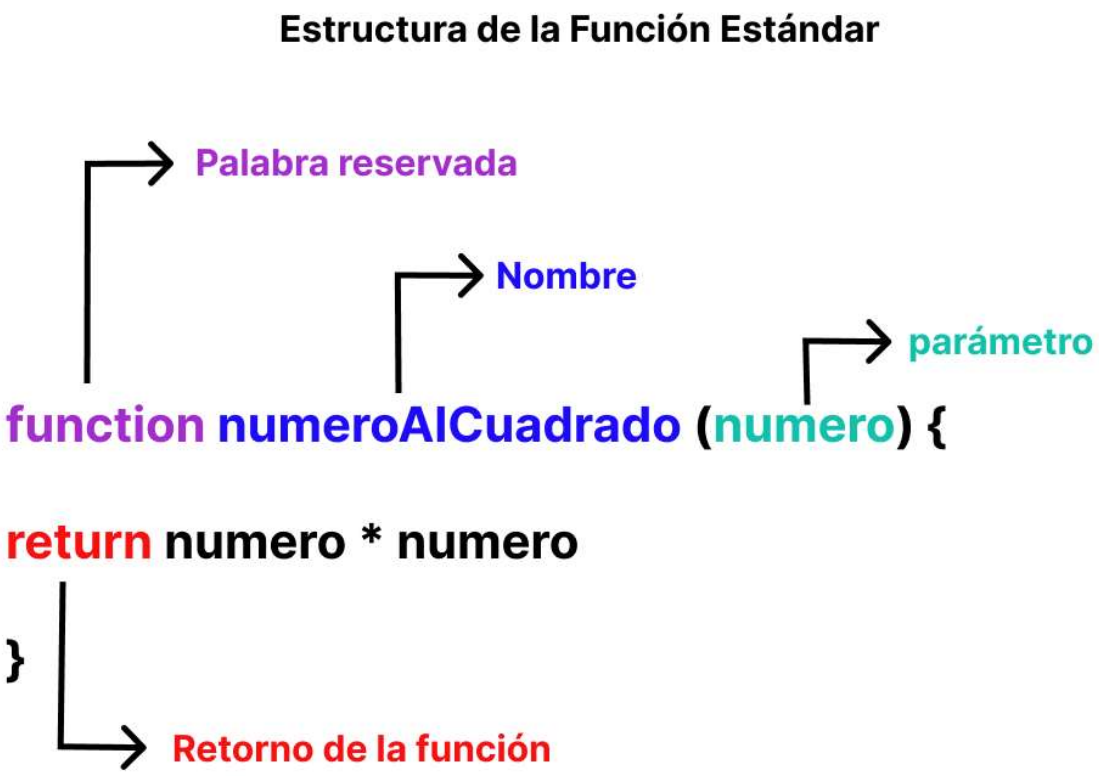
- **Función estándar;**

- **Funciones de Flecha (Arrow Functions);**
- **Funciones dentro de eventos y métodos de array.**

Estructura de una función estándar

Para declarar una función, debe usar la palabra clave `function` (en-US), seguida del nombre de la función, el parámetro de la función, cerrados entre paréntesis `()`. Y, las declaraciones de JavaScript que definen la función, cerradas entre llaves `{}`.

Vea este ejemplo de código con utilización de una función llamada `numeroAlCuadrado`:



Comprendamos ahora la estructura de esta función y cómo funciona, analisando cada elemento presente en ella:

La función se inicia usando la *palabra clave* `function`. En JavaScript, no podemos usar palabras que estén reservadas para el lenguaje, como nombrar objetos. Esto se debe a que se interpretan al generar el código binario (que traduce la instrucción, permitiendo que la máquina entienda lo que se solicitó).

Si, no conoce las palabras reservadas de JavaScript, no se preocupe porque no necesitamos memorizar todas las palabras para aprender a programar, y definitivamente se acostumbrará a ellas a medida que practique.

Después de usar la palabra reservada, nonbramos nuestra función, denominada `numeroAlCuadrado`, con la primera palabra comenzando con letra minúscula y la segunda con mayúscula sin espacio entre palabras. A esta práctica le conocemos como *CamelCase*, para obtener más información haz clic aquí.

Después de nombrar la función, proporcionamos un `parámetro:numero`, en ese caso. Ese parámetro recibió un argumento, que son datos que se enviarán cuando se llame a la función. Un parámetro puede tomar, por ejemplo, un número o un String como argumento.

En este caso, si ingresamos el número 5 como argumento, la función tomará el valor pasado y lo insertará en el parámetro `numero`. Con eso, dentro de la función, tendremos **numero = 5**. A partir de esta asignación, la función calculará y devolverá el resultado de 5 al cuadrado.

Retorno de función

La palabra clave `return` define el valor devuelto por una función. Cuando no se especifica ningún valor, se devolverá `undefined`. Además, *return* también interrumpe la ejecución de la función actual.

- **Retorno con valor**

Cuando especificamos un valor, el retorno será el valor especificado. Como, en este ejemplo:

```
function devuelveValor() {
    return 1;
}
```

El código anterior devuelve `1` como resultado.

- **Retorno sin valor**

Cuando no especificamos un valor, el valor devuelto es `undefined`, que es, como su nombre lo indica, un valor indefinido. Mira este ejemplo:

```
function retornoVacio() {
    return;
}
```

El código anterior devuelve `undefined` como resultado.

- **Retorno de más de un valor**

Es posible que una función tenga más de un `return` aplicado a su estructura y esto puede suceder cuando queremos controlar con mayor sensibilidad el flujo de una ejecución. Podemos realizar cambios en los "estados" en una variable *booleana* `true/false`, actualizando su valor y devolviéndolo dentro de estructuras condicionales, por ejemplo:

```
function verdaderoOFalso( datos ){
    if ( datos.length > 0){
    return true
}else{
    return false
}
```

Funciones de Flecha (Arrow Functions)

Disponible desde la actualización de 2015 a la versión 6 de JavaScript (ES6), las funciones de flecha son casi siempre funciones anónimas y buscan simplificar la ejecución, utilizadas por métodos de *array* facilitando retornos simples. Para entender en la práctica vamos a comparar la sintaxis de una función estándar y de las funciones de flecha para destacar sus diferencias:

```
//Función 1 (estándar)

function suma(a,b){
    return a + b
}

//Función 2 (función de flechas)

const sum = (a, b) =>{
    a + b
}
```

Tenemos en el siguiente ejemplo una función estándar de suma que realizará la adición de los parámetros `a` y `b`, y como ya hemos visto hasta ahora tenemos el uso de la palabra reservada `function`, el `nombre`, sus `parámetros` y su `retorno`; En la función 2 tenemos ahora un ejemplo de función flecha, donde **no es** necesario usar la palabra reservada `function`, y en este caso la asignamos a una constante o variable en caso de que **no sea anónima**.

Después de declarar nuestros parámetros utilizamos la flecha, compuesta por un símbolo de igual = seguido de un mayor que > y abrimos el scope de nuestra función, donde la palabra return no es necesaria en el caso de funciones con un solo argumento. En este caso decimos que su retorno es **implícito**, o sea, todo lo que está después de la flecha es tomado como return por JavaScript.

Podemos simplificar la sintaxis creando la función en el patrón *inline*, que es cuando toda la función se escribe en una única línea,tomando nuestro ejemplo de la función 2 anterior y quitando los corchetes { }. Ya que, como se ha mencionado, después de la flecha =>, se asumirá que todo es retorno.

```
//ejemplo en una línea

const suma = (a, b) => a + b  //función de flechas inline (en una línea)
```

Recordando que ninguna simplificación es obligatoria, dejando al programador la decisión de qué declaración utilizar en cada momento de la escritura del código.

Funciones dentro de Eventos con Arrow Functions y Métodos de array

Las funciones de flecha no se crearon sólo para economizar unas líneas de código y con el siguiente ejemplo veremos más de las diferencias de usos en la programación.

```
//Función 1 (evento)

//selección de elementos del HTML
const boton = document.querySelector('button');

boton.addEventListener('click', () =>
    console.log("el botón ha sido clicado")
)
```

Como mencionamos, las funciones de flecha se utilizan mucho para eventos y para simplificar las ejecuciones. En el ejemplo anterior empezamos capturando un elemento existente en HTML y utilizamos el querySelector para usarlo como una variable en JavaScript, más información clic aqui.

A continuación, utilizamos nuestra variable que contiene el selector `button` para disparar un evento (addEventListener) que ejecutará una acción de click. Un ejemplo clásico que nos ayuda mucho a entender la lógica, después del argumento 'click', que será el tipo de nuestro evento, tenemos una coma para separarlos (igual que separamos los parámetros de las funciones estándar).

Un conjunto de paréntesis vacíos que representan exactamente una función anónima que realizará la acción de mostrar en la consola el mensaje de que el botón ha sido clicado. De esta manera tenemos una ejecución completa sin necesidad de una función externa que ayude en esta ejecución y que puede ser utilizada de muchas maneras diferentes.

```
//Función 2 (evento com función de flecha anónima

nombres = ["Jhon", "Sebas", "Ana"]

nombres.forEach(element =>
    console.log(element)
);

//Array(3) > "Jhon", "Sebas", "Ana"
```

Siguiendo con el ejemplo arriba, empezamos declarando un array de nombres que usaremos en la siguiente función que ejecutará el método forEach. Que de manera sencilla, realiza un bucle (como el bucle for o while) para pasar por cada elemento del array de nombres, aprende más haciendo clic aquí.

Para esto declaramos primero el array que será recorrido, el método forEach y ejecutamos nuestra función de flecha. Ahora diciendo que el parámetro element pasará de manera directa por cada elemento del array y con retorno directo ejecutará la tarea de imprimir en la consola, cada uno de los nombres, teniendo un resultado como el que se demuestra.

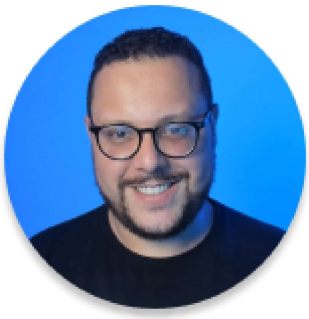
Ambas funciones pueden asociarse para ejecutar un evento que genere una acción para el usuario y el conocimiento de los distintos tipos de funciones le dará un mayor control sobre su gestión de los comandos que se ejecutarán dentro de JavaScript.

Ahora conoces con más detalle no sólo los distintos tipos de funciones sino también como JavaScript interpreta cada una de ellas y las ejecuta parcialmente según sus instrucciones. No dejes de consultar otros artículos para aprender mucho más sobre JavaScript y sobre otros temas relacionados con la programación.

Sobre los autores:

Marianna Costa

Brasileña, estudiante de ingeniería informática y formo parte del equipo Scuba Team en Alura Latam. Soy amante de la tecnología desde siempre y me enamoré del desarrollo front end. Soy muy curiosa y siempre estoy interesada en aprender sobre los diversos asuntos. Me encanta compartir y recibir conocimiento porque creo firmemente que *"compartir conocimiento es lo mismo que difundir desarrollo".*



Wesley Bastos

Brasileño, instructor y desarrollador de front-end Fascinado por los temas de JavaScript/CSS y por enseñar cómo empezar la carrera de desarrollador. Economista de formación y mis principales aficiones son estudiar diseño y jugar al rugby. Creo que aprender sobre tecnología cambia vidas y abre puertas en uno de los mercados más importantes de hoy en día.

ARTÍCULOS DE TECNOLOGÍA

un solo pago de US$49,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

ANUAL

US$79,90

un solo pago de US$79,90